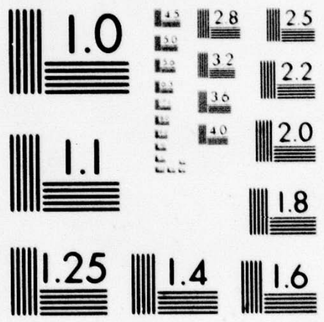
AD-A075 181 FOREIGN TECHNOLOGY DIV WRIGHT-PATTERSON AFB OH F/G 9/2
STRUCTURED PROGRAMMING AND THE TECHNIQUE OF DEVELOPING PROGRAMM--ETC(U)
APR 79 R JANKOVIC
UNCLASSIFIED FTD-ID(RS)T-2286-78 NL

1 OF 1
AD
A075181

END
DATE
FILMED
11-79
DDC


1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

FTD-ID(RS)T-2286-78

FOREIGN TECHNOLOGY DIVISION

AIR FORCE SYSTEMS COMMAND

STRUCTURED PROGRAMMING AND THE TECHNIQUE OF DEVELOPING PROGRAMMING SYSTEMS

by

Radomir Jankovic

FOREIGN TECHNOLOGY DIVISION

EDITED TRANSLATION

FTD-ID(RS)T-2286-78 5 April 1979

MICROFICHE NR: FTD-79-C-000523

STRUCTURED PROGRAMMING AND THE
TECHNIQUE OF DEVELOPING
PROGRAMMING SYSTEMS

By: Radomir Jankovic

English pages: 17

Source: Nauchno-Tehnicki Pregled, Volume 27,
Number 1, 1977, pages 49-55

Country of origin: Yugoslavia
Translated by: Linguistics Systems, Inc.
F33657-78-D-0618
Michael Peter Bakich
Requester: FTD/TQCS
Approved for public release;
distribution unlimited.

STRUCTURED PROGRAMMING AND THE TECHNIQUE OF DEVELOPING PROGRAMMING SYSTEMS

By

Radomir Jankovic

The sudden and over all development of computational technique , particularly in programming systems, has brought with it a series of accompanying problems which have made classical techniques for devising programs and their proofs ineffectual. As a result of the ideas and the work of others who strive to overcome these difficulties, I became acquainted with structured programming. This new technique for developing a complex system of programming is described along with basic ideas about the corresponding organization of the task of programming. Described are basic control structures in structured programs, and also pointed out, in connection with the characteristics of several typical programming languages, is the possibility of applying the technology of structured programming.

1. Introduction

Today, a complete and clear answer still can't be given to the question, what is structured programming?

The basic reason for this is that structured programming, as a new technique for the development and organization of a programming system, appeared relatively recently. Because of this, the picture was a mixture of many different ideas and directions in research and in practice. It is not possible to reliably assert who the inventor of structured programming really is because it is,beyond all doubts, the product of the work of many men who sought different ways out of and in solution to the problems introduced by the sudden and widespread development of computational techniques and, especially, programming systems.

Nevertheless it is thought that the basic idea comes from E. W. Dijkstra of Holland, in reference to his article published in March 1968 (1), in which the author brought forth his position on the dangers of using the command for an unconditional jump, in its classical form, GO TO, when writing programs for higher programming languages. Namely, he thought that, along with complicated control paths and an abundance of superfluous marks which,altogether, contribute to an unintelligible program text, this instruction makes complete an insurmountable difficulty when trying to establish paired coordinates which have to contain information about the text until the process, which is developed under the control of the concerned program, succeeds. On the other hand, basic program structures, like sequential structure, conditional structure (<u>if</u> B <u>then</u> A), alternative structure (<u>if</u> B <u>then</u> A1 <u>else</u> A2),and repetitive structure (<u>while</u> B <u>repeat</u> A, that is, <u>repeat</u> A <u>until</u> B), makes possible the establishment of this type of paired coordinates with which itispossible to describe the progression of the process. Dijkstra later also published a book (2), precisely stated, a textual index, in which he delineates his views on the methodology of programming. This methodology, which, in the development of the program step by step, along with exactly proving the correctness of each step on every level of abstraction, from the general supposition of the problem to providing valuable instructions in the programming language, has to provide for getting the most correct and clear program possible in order to achieve that which is intended.

These works, which still, more or less, remain in a theoretical framework, have been greatly interesting due to the results explained by IBM in connection with their work for the New York Times.

This work is not based, in its entirety, on structured programming but, also, on the concept of what is called the "Chief Programmer Team," authored by F.T. Baker and Harlin Mills. Inherent in this concept is the idea that large programming systems are to be developed by small, well organized and managed teams consisting of an experienced programmer and the remaining necessary personnel, along with the use of the special techniques of structured programming. Meanwhile, without considering the good organization of both the work and the accompanying process of coding the programs, the author's concepts emphasize those parts of structured programming crucial to the making of program systems of a high degree of reliability.

This article will be an honest attempt to conditionally define the concept of structured programming. It is, generally speaking, a new access to programming and the organization of programming systems, as well as to the organization of the process of their construction. The goal of this concept is to increase the reliability of and to reduce the expenses of the programming system in development. This concept applies as well to the maintenance of the system. As such, structured programming can be, for the most part, reduced to the following:

1. the joint use of several simple program structures for control paths in the program,

2. a "Top-down" approach to the compilation of the program and, therefore, similarly, to the text (in the language spoken of), where there are no jumps backward and, generally, no complicated unconditional jumps, and

3. modularity and hierarchy in the organization of large programs.

2. Problems In The Development Of Large Programming Systems

Structured programming, disregarding that which is explained in several places and by several people, is the result of difficulties encountered in the development of large programming systems. That is, difficulties not only in the sense of the project finishing on time, but also that, generally, suitable programs must be produced which can fill the stated requirements.

The usual classical technique of programming and tests of a program (writing of the program--hand testing by different programmers--a trial run of the program on the computer with test data for which is expected a known result--the searching for and correcting of noticeable errors--repeated tests of the program done on the computer, etc.) were demonstrated, in the case of a program of these demensions, to be completely ineffective.

With the compilation of this type of program, the programmer, daily, codes five to ten debugged instructions. As it is, just coding such a group of instructions takes several minutes. It is apparent that the programmer spends the largest part of his working time eliminating errors. As opposed to with difficult problems, with the smaller programs, this sort of thing had not been noticed. There was no hesitation then about just programming. This, of course, is without considering that it was also then clear that there existed programmers of varying degrees of skill. Before the compilation of the program, a primary requirement is stated--that you have to carry out the assignment of the function, and also a secondary requirement--that you have to be effective in the sense of having the necessary hardware capacity and time.

Meanwhile, we now come to the reorientation of the secondary requirement which is now reduced to the following three factors:

1. dependability of the programming system,
2. maintenance,
3. feasibility of modification and enlargement.

These factors express themselves through the expenses that surround everything with which they are connected.

It was noticed that with the growth of the programming system there was also a sudden growth of expenses surrounding their maintenance, and especially surrounding the coding of modifications and the elimination of errors.

On the other hand, every coded modification may result in a string of new errors and the creation of hard to understand control paths without consideration whether those changes were made earlier by the author of the program or some new person. Frequently the effort which is put into that work is so large that it is given up and the program begun again or, as a result, it will give an undependable end product.

Structured programming increases the clearness and understandability of the program that is obtained, in the first place, through the detailing of the command routes using simple basic structures, in addition to writing the program, keeping in mind the original text, without unclear, unconditional jumps. This will prove convenient for the reading and understanding of the program. This is good, also, for the economical use of space in the operational memory of the computer. A program in the computer has to be recorded when the program is used and, with this method, the finished parts of the program are eliminated and space is freed. Finally, the modular hierarchical organization of large programming systems is also utilized in order that the elementary program modules will be short (and within a persons capability to percieve and understand), and the connections between them clear and unambiguous.

3. Basic Control Structures

As for the theoretical basis for the idea of developing a program with the help of a group of simple, basic control structures, the article generally cited is that published in May of 1966, by C. Bohm and G. Jacopini. They showed that any program could be written with the help of the following three basic control structures (Fig. 1):

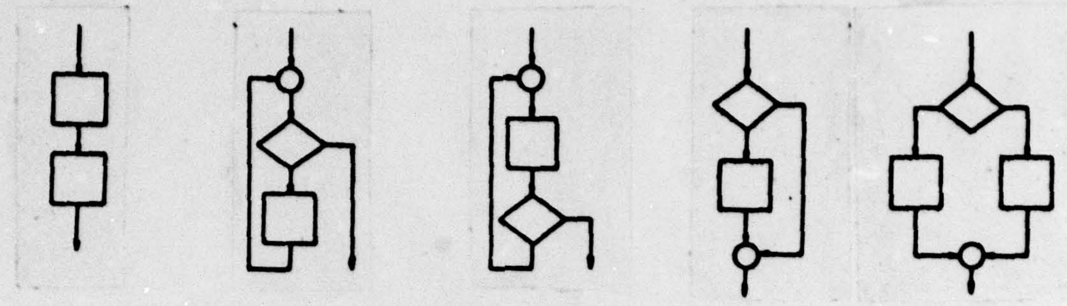


Fig. 1—Basic Control Structures

1. Sequential structure, by which the program instructions are carried out according to the order in which were written.

2. Selective structure, by which it is decided which following activity will be undertaken under the control of the program in question. In the scope of this structure two cases can be explained. These are, simply, conditional structures of the form (if B then A) and some compounds of this, and alternative structures of the form (if B then A1 else A2).

3. Repetative structure, by which is explained the return mechanism. This mechanism utilizes a loop in the program instructions and is finished when some sought for condition is filled. This accomplishes an iterative procedure. This structure is, in general, explained in its two basic cases by the forms, (while B repeat A) and (repeat A until B).

Every algorithm, itself, through a procedure called normalization, to the degree it was presented in the form of a flowchart , may duplicate, in itself, an equivalent algorithm, that is, a block diagram (equivalent in the sense of completing all functions as does the original algorithm) consisting of the elementary control structures in Fig. 1.

However much the programming language at one's disposal contains instructions which perform the control functions of these elementary structures, to that degree can one write programs using the techniques of structured programming already given. Using this, one can realize all of the benefits of a structured programming approach.

Harlan Mills expands these results with requirements that program modules, within the limits of just this sort of control structure, have only just one entry and exit point. In his article, published January 1975 (4), it was shown that, if the program is written with this organization, its correctness can be proved. To this end, also given was the theory for proving the correctness of basic program control structures.

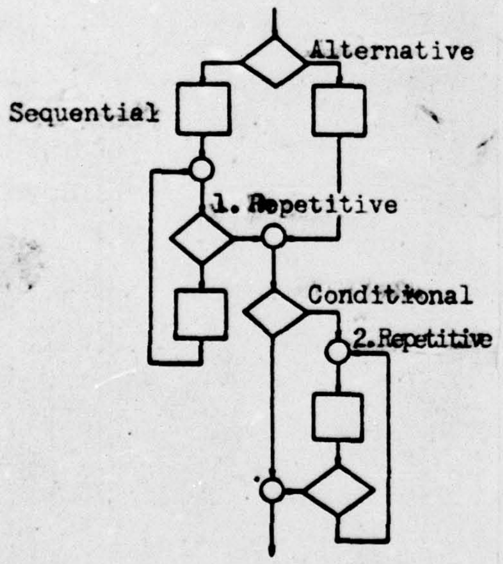


Fig. 2—Complex control structure

With the use of this structure, it is possible to write the program organized from the top toward the bottom (top-down), in which there are no backward jumps and the classical command of an unconditional jump (GO TO) becomes superfluous. Signs in the program are also unnecessary. Control structures may develope within the framework of other control structures (Fig. 2), but they, also, continue to retain features that have just one entry and exit point. The exception to this rule illustrates generalizations of selective structures of the form, (<u>case</u> i <u>of</u> S1; S2; S3; ...; Sn) and the occurence of an abnormal finish of a repetitive structure (Fig. 3). This latter exception can be very effective in individual situations that may crop up in the daily routine of programming and may be used with an eye towards saving both space and time.

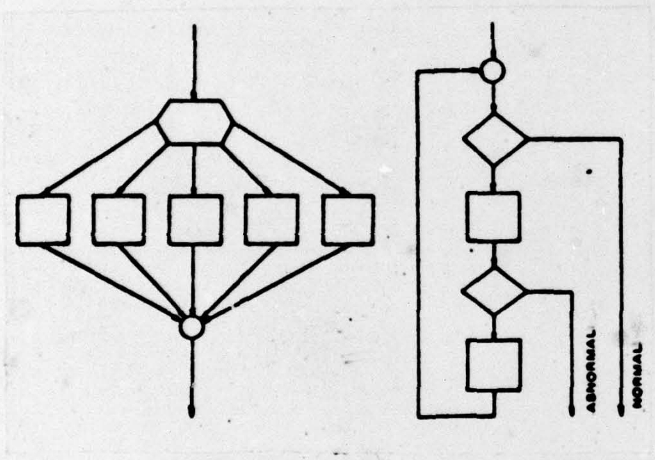


Fig. 3--A generalization of selective and abnormally terminated repetitive structure.

In connection with the production of a flow chart in which there has to be presented an algorithm in a form suitable for structured programming, a special technique is described in an article, published in May 1975, by J.B. Holton and B. Bryan. Three basic control structures illustrating this technique are shown in Fig. 4. It is shown that sequential structure can be accomplished with simple linear modules consisting of steps 1, 2 and 3. Number 1, in the upper right hand

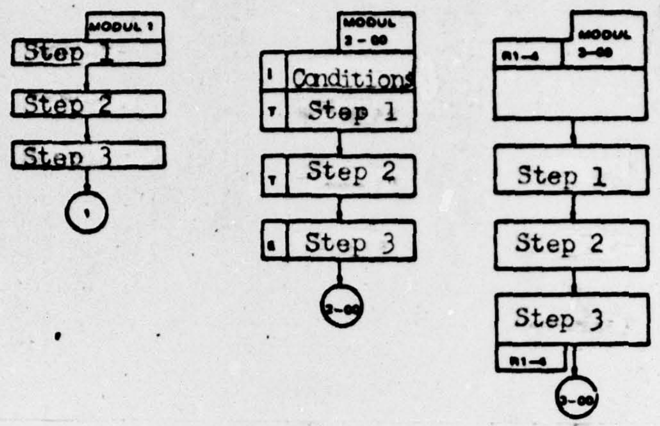


Fig. 4--Schematic of a flowchart

corner of the first set, designates the start of that module in the framework of the entire flowchart set. The same number, in a node at the end of the module, designates its end. Selective structure, and its more complicated cases, is also described(with respect to the modules of steps 1, 2 and 3). In this case, the letter "I" identifies the condition, or conditions, which have to be questioned. The letter "T", beside steps 1 and 2, indicates that it will, itself, finish these two steps if the cases in question are satisfied. Finally, the letter "E", next to step 3, says that this step will be completed if the cases in question aren't satisfied. In a similar manner, repetitive structure is also described. It consists, here, of four process blocks. The first block represents the repetition delimiter which pertains to the exit condition from the loop. The following three blocks describe steps 1, 2 and 3 which are completed in the loop.

As far as the programming language, which has to be applied in connection with structured programming, is concerned, it is desireable to utilize ALGOL and

PL/1, because they already contain structures favorable for the application of the described techniques. Unfortunately, the language which the widest group of programmers know, FORTRAN, isn't very suitable for the application of structured programming. This, meantime, does not mean that the application of the principles of structured programming will not give good results in spite of large programs written in FORTRAN. In his article, published in November 1974 (6), David Gries says: "...We must distinguish between the programming language and the notation used while programming. One doesn't program in a programming language but into it. We use any notation which fits the problem; but we may need gotos to"simulate" that notation in the programming language."

Considering that, in the language of FORTRAN, there are instructions which are useful for the simulation of singular control structures necessary for the structuring of programming, there is absolutely no reason that that technique should not be used in cases when there is only FORTRAN at ones disposal. This would be done with the expectation of perceptable improvements in the effect of FORTRAN on classical programming. Referred to are instructions for the unconditional jump (GO TO), both types of IF instructions (logical and arithmetical), jump instructions with cross checks, instructions for the DO loop, etc.

4. Modular Development Of The Program "Top down"

An important aspect of structured programming, as a technique for the development of complex program systems, is that it also establishes a modular, heirarchical access developing the program top down. Namely, just writing the program without the GO TO command will not include many improvements if the more complex program structures aren't developed in a fixed sequential heirarchy of organized, simple structures--of program modules which announce sequential functions in the system.

Access to the development of the program system top down shows potential in that it can approach the development of the system from the highest level, and , as it is with respect to the development of a program, so to in the implementation of the program. Simpler program-modules are immediately plugged into the system. This is done so that they, while they still aren't developed, replace simple groups of instruction which simulate their functions in the framework of the system. Afterwards, the program modules are developed, tested and implemented in the system. This is done through the hierarchy of upper to lower levels, top down, until the simplest program modules, which perform the various functions in the system, are implemented in it.

Program modules, in this type of organization,have to be short enough in order that a person may be able to grasp them and, too, the correctness of the program must, in the least, be easily proven if it was not already obvious. In ~~pratical~~ practical terms, this means that the program module cannot be longer than the set of instructions that go on a single sheet of the computer print out or on one screen of the visual display.

The significant advantage of this method of developing complex program systems is its high dependability, in the sense that, when the operation is finished, the program will truly fill the stated requirements. For just that reason, simulators of singular, still undeveloped, untested and unimplimented program modules, which perform various functions in the framework of the system, are immediately plugged in, in reference to testing whether the solution is in order and correct. In this respect, it is possible to perform an overall check of the accuracy of the conceptions of the entire project relatively quickly. The design of the entire system is developed in hierarchical levels and, when considering just one level in reference to the operation of all program modules on that level tested, the number of implementation iterations is reduced to one. That means that it is no longer a necessity to ever guess about that level. This would hold true throughout the entire project.

Here is, surely, in the entire program system, literally the primary principle for the development of a program, step by step, from the highest to the lowest level of abstraction. The difference is in that, here, the final goal is to get to organized blocks of accurate elementary programming modules, while in the development of a program, the final goal is to get blocks of organized instructions in the program language. This,of course, depends on the available computer.

5. The Organization Of A Programmer Team

The problems which lead to structured programming, as a convenient technique for developing and organizing large program systems, also resulted in innovations in the organization of the work toward the development of these types of projects. One of the most important contributions in this review is the concept of "The Chief Programmer Team," by F.T. Baker and Harlan Mills (7).

This concept introduces a new approach to the organization of the operation of the programmer team. This new approach was developed, possibly, thanks to obvious excesses in the technique of programming. In this approach, the breakdown of the entire operation surrounding programming into specialized operations is carried out, the mutual relations of specialists are defined, the possibility of necessary inspections by specialists in the project being developed is achieved and the training and career development of this type of personnel is made possible. The chief programmer team, in terms of a classical programmer group where the work was simply shared between multi-purpose programmers, isn't plagued with functional limitations, a lack of discipline and a lack of team work. This results in large increases in the productivity and quality of programming.

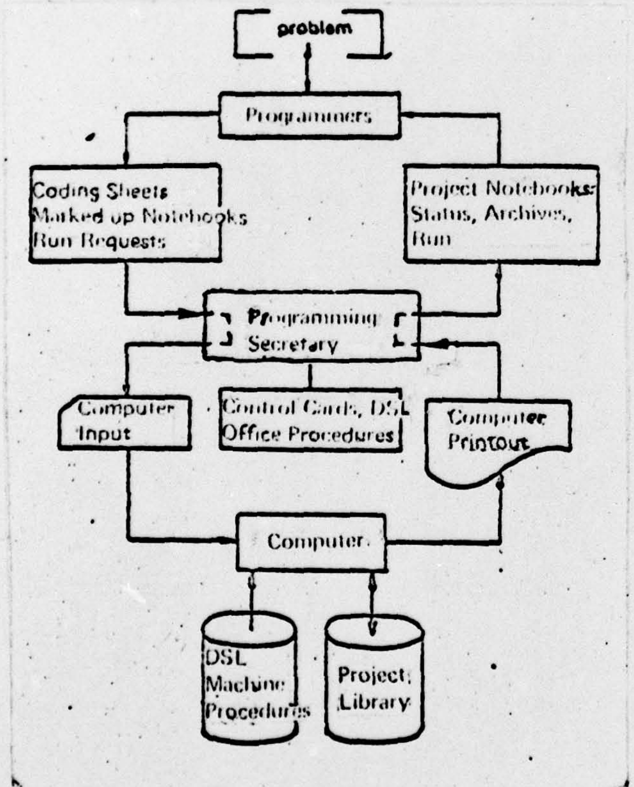


Fig. 5—The Development Support Library and its relationship to the people in the programmer team.

Besides this, the operations of the chief programmer team rely on two important innovations. The first of these is the already described technique of structured programming. The second consists of the establishment of the "Development Support Library," whose basic organization and relation to the staff in the programming team is shown in Fig. 5. With the aid of this library, all programs in development are maintained in obvious, visible, standardized forms.

The nucleus of a chief programmer team consists of a chief programmer, a backup programmer and a programming secretary. The demand for additional personel is defined by the chief programmer, depending on the size of the project which the team has to carry out. The average team will have from three to five programmers, one secretary and, also according to needs, other specialists. This type of team will be able, independently, to realize a project of 100,000 instructions of orriginal code, while larger projects will require the enlargement of the organization to a group of teams.

The chief programmer controls the entire project and everyone on the team reports to him directly. He also programs the most important and most critical parts of the program system and, therefore, those parts which on higher levels connect and organize the operation of the remainder of the program modules. He specifies the program requirements for the rest of the programmers for the production of some of the program modules. He then reviews these and incorporates them in the system as they are ready.

The following person in the hierarchy of the chief programmer team has to be totally familiar with the project in development. The backup programmer, therefore, participates in making all important decisions and, in that way, must be ready to assume the leadership role in the team, if required. He also participates in the design of the system and in coding parts of the program system under the direction of the chief programmer. In addition to that, he is responsible for the research work in the team, he works at selecting the best program stratagy,and, in this same way, independent of the chief programmer, is in charge of planning tests of the system. The programming secretary has the chief responsibility of maintaining the Development Support Library and, in that sense, prepares the computer entries and the products of the computer runs, as well as all other necessary notations and documentations, concerning the standing of the program, which are found in the development of the program, as well as in the development of the program system in its entirety.

#

Bibliography

[1] Dijkstra,E.W.: Go to Statement Considered Harmful , CACM, March 1968, 147–148.

[2] Dijkstra,E.W.: Notes on Structured Programming , u okviru Structured Programming , Dijkstra, Dahl, Hoare, Academic Press, London, 1972.

[3] Böhm,C.,Jacopini,G.: Flow Diagrams, Turing Machines and Languages With Only Two Formation Rules CACM, May 1966, 366–371.

[4] Mills,H.D.: The New Math of Computer Programming , CACM, Jan 75, 43–48.

[5] Holton,J.B.,Bryan,B.: Structured Top–Down Flowcharting , Datamation, May 1975.

[6] Gries,D.: On Structured Programming , CACM, Now 74, 655–657.

[7] Časopis Datamation, Dec 1973., specijalno izdanje posvećeno struktuiranom programiranju.

[8] Wirth,N.: Program Developement by Stepwise Refinement , CACM, April 1971, 221–227.

[9] Bauer,F.L.: Materijali sa simpozijuma Informatica 74, Bled.

DISTRIBUTION LIST

DISTRIBUTION DIRECT TO RECIPIENT

ORGANIZATION	MICROFICHE	ORGANIZATION	MICROFICHE
A205 DMATC	1	E053 AF/INAKA	1
A210 DMAAC	2	E017 AF/RDXTR-W	1
B344 DIA/RDS-3C	9	E403 AFSC/INA	1
C043 USAMIIA	1	E404 AEDC	1
C509 BALLISTIC RES LABS	1	E408 AFWL	1
C510 AIR MOBILITY R&D LAB/FIO	1	E410 ADTC	1
C513 PICATINNY ARSENAL	1	FTD	
C535 AVIATION SYS COMD	1	CCN	1
C591 FSTC	5	ASD/FTD/NIIS	3
C619 MIA REDSTONE	1	NIA/PHS	1
D008 NISC	1	NIIS	2
H300 USAICE (USAREUR)	1		
P005 DOE	1		
P050 CIA/CRB/ADD/SD	2		
NAVORDSTA (50L)	1		
NASA/KSI	1		
AFIT/LD	1		
LLL/Code L-389	1		
NSA/1213/TDL	2		